

(12)

Technical Report

622

LBS — Lincoln Boolean Synthesizer

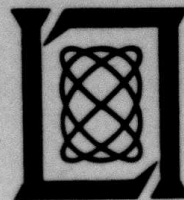
J.R. Southard
A. Domic
K.W. Crouch

1 September 1982

Prepared for the Defense Advanced Research Projects Agency
under Electronic Systems Division Contract F19628-80-C-0002 by**Lincoln Laboratory**

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LEXINGTON, MASSACHUSETTS



Approved for public release; distribution unlimited.

*Original contains color
plates: All DTIC reproductions
will be in black and
white*

DTIC
ELECTE
S NOV 02 1982 **D**

E

82 11 02 044

DTIC FILE COPY

ADA 120999

The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. This work was sponsored by the Defense Advanced Research Projects Agency under Air Force Contract F19628-80-C-0002 (ARPA Order 3797).

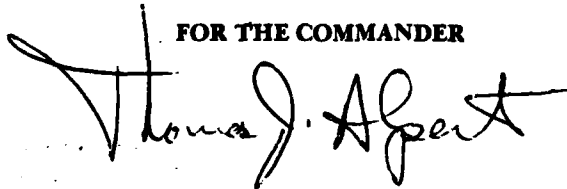
This report may be reproduced to satisfy needs of U.S. Government agencies.

The views and conclusions contained in this document are those of the contractor and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the United States Government.

The Public Affairs Office has reviewed this report, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

A handwritten signature in dark ink, appearing to read "Thomas J. Alpert". The signature is stylized with a large, sweeping initial "T" and a long, horizontal stroke at the end.

Thomas J. Alpert, Major, USAF
Chief, ESD Lincoln Laboratory Project Office

Non-Lincoln Recipients

PLEASE DO NOT RETURN

Permission is given to destroy this document
when it is no longer needed.

ERRATA SHEET
for
TECHNICAL REPORT 622

An error has been detected in Figure 2 (page 6) of MIT/LL Technical Report-622 ("LBS — Lincoln Boolean Synthesizer," by J.R. Southard, A. Domic, and K.W. Crouch, dated 1 September 1982). Please insert the attached, corrected version of Figure 2 into your copy of that report.

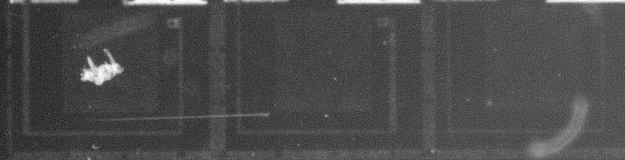
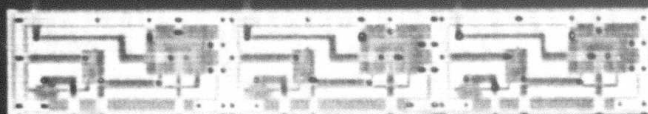
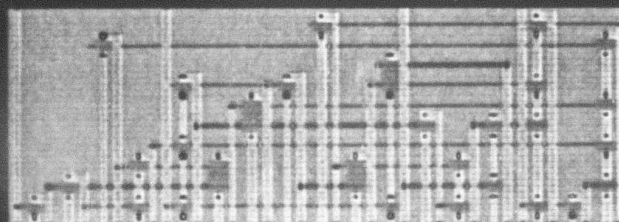
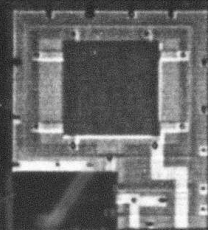
The views and conclusions contained in this document are those of the contractor and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the United States Government.

The Public Affairs Office has reviewed this report, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

Approved for public release; distribution unlimited.

21 September 1982

Publications
M.I.T. Lincoln Laboratory
P.O. Box 73
Lexington, MA 02173-0073



MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

LBS — LINCOLN BOOLEAN SYNTHESIZER

J.R. SOUTHARD
A. DOMIC
K.W. CROUCH
Group 24



Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

TECHNICAL REPORT 622

1 SEPTEMBER 1962


"Original contains color plates: All DTIC reproductions will be in black and white"

Approved for public release; distribution unlimited.

LEXINGTON

MASSACHUSETTS

ABSTRACT



The Lincoln Boolean Synthesizer (LBS) creates custom integrated circuits from boolean logic equations. The currently supported integration technology is a fully customizable CMOS process with 5 micron channel width, P-tub and, one level of metal. This document describes the capabilities of LBS and provides essential information for operating LBS and an LBS simulator which has also been developed.




TABLE OF CONTENTS

ABSTRACT	111
CONTENTS	v
1. INTRODUCTION	1
2. LBS SPECIFICATION	1
2.1 Boolean Expressions	1
2.2 Outputs and Internals	2
2.3 Observation on Efficiency	2
3. INTERFACE TO THE OUTSIDE WORLD	3
4. THE NAME OPERATOR	3
5. AN EXAMPLE	4
6. OPERATION	4
7. SIMULATOR	7
REFERENCES	9

1. INTRODUCTION

The Lincoln Boolean Synthesizer (LBS) creates custom integrated circuits from boolean logic equations. The currently supported integration technology is a fully customizable CMOS process with 5 micron channel width, P-tub and, one level of metal. A simulator is provided.¹

2. LBS SPECIFICATION

An LBS specification consists of a set of boolean logic equations called a "network." The network will specify input, output, and logic. Network inputs and outputs are so-called "named" items. As such, they can be simply referenced in the logic specification. Network inputs are driven by external circuitry and network outputs drive external circuitry. When an LBS generated circuit is fabricated, inputs and outputs will be bonded to leads of the chip.

2.1 Boolean Expressions

The backbone of LBS specifications is the boolean expression. A boolean expression consists of an operation and a list of arguments, viz:

(or a b)

This expression will construct circuitry (called a "gate") to or the items named a and b . Other supported operators are nor, and, nand, not, and xor. Arguments may be named items such as network inputs, or they may be other expressions, viz:

(or a (xor c d)).

¹ An up-to-date copy of this document is available on the Lincoln Laboratory RVLSI VAX computer as /usr/local/doc/lbs.

In this case a gate will be constructed to evaluate the xor of c and d. Another gate will or that result with a. There is no practical limit to this nesting or cascading of operations. Incidentally, all the operators except not accept any number of arguments, not just two.

2.2 Outputs and Internals

The results of boolean expressions can be named as an output of the network:

```
(out a (or b c))
```

This creates a network output named a which is attached to the gate evaluating the or of b and c. Note that since it is an output, a is a named value and can be used as such in other logic expressions.

Sometimes it is useful to create a named value which is neither an input nor an output of the network. This name will be referenced internally, somewhat like a local variable in a subroutine.

```
(setq a (or b c))
```

creates a named value which is neither input nor output but can be used in other logic expressions as though it was.

2.3 Observation on Efficiency

Let us consider the two networks:

```
((setq temp (nor a b))  
 (out o1 (xor temp c))  
 (out o2 (xor temp d)))
```

and

```
((out o1 (xor (nor a b) c))  
 (out o2 (xor (nor a b) d)))
```

Logically these are equivalent; however, it would seem that by using `temp` the first specification creates 3 gates (two `xors` and one `nor`), while the second specification creates 4 gates (two `xors` and two `nors`). There is an interesting analogy here between this IC design example and similar controversy between standard procedural languages and the so-called "applicative" or "functional" languages. What actually happens is that LBS works a little harder on the second specification, detects that two identical `nor` gates are being called for, and generates the same layout for both specifications.

3. INTERFACE TO THE THE OUTSIDE WORLD

In the present version of LBS, input and output leads will appear on opposite sides of the chip. The order of the inputs and outputs both depend on their order or appearance in the specification. Inputs can be ordered independently of their appearance in `out` and `setq` equations by use of an expression containing only the input name. For example, the network

```
((out bnorc (nor b c))
 (out nota (not a)))
```

would order the inputs `b`, `c`, and `a`. However, the network

```
(a
 b
 c
 (out bnorc (nor b c))
 (out nota (not a)))
```

would order the inputs `a`, `b`, and `c`. There would be no difference in the number of gates.

4. THE NAME OPERATOR

The obsolete operator `name` is supported for historical reasons. It is

documented so that very old examples may be understood. Anywhere that a "named" item appears in a boolean expression a `name` expression can be substituted. For example:

```
(or a (xor c d))
```

is equivalent to

```
(or (name a) (xor (name c) (name d)))
```

5. AN EXAMPLE

The following code creates a master-slave flip-flop.

```
((setq om1 (nor (and (not in1) phia) om1bar clear))
 (setq om1bar (nor (and in1 phia) om1))
 (out os1 (nor (and om1bar phib) os1bar clear))
 (setq os1bar (nor (and om1 phib) os1))
 (setq phia (nor clock phib
                (and om1bar phib)
                (and om1 phib)))
 (setq phib (nor (not clock) phia
                (and (not in1) phia)
                (and in1 phia))))
```

The equivalent logic diagram is Fig. 1. A two phase clock is generated internally from the single clock input. The master flip-flop tracks the input during `phia`, and is fixed during `phib`. The slave is fixed during `phia`, and transfers the output of the master during `phib`. The circuit which internally generates two phase clock (`phia` and `phib`) was synthesized by writing down the logic expression for figure 7.7 in Introduction to VLSI Systems by Mead and Conway [1]. Figure 2 is the resultant layout.

6. OPERATION

Make sure that your \$path includes /usr/vlsi/bin. If you do not understand this, please contact someone who knows something about UNIX. Create

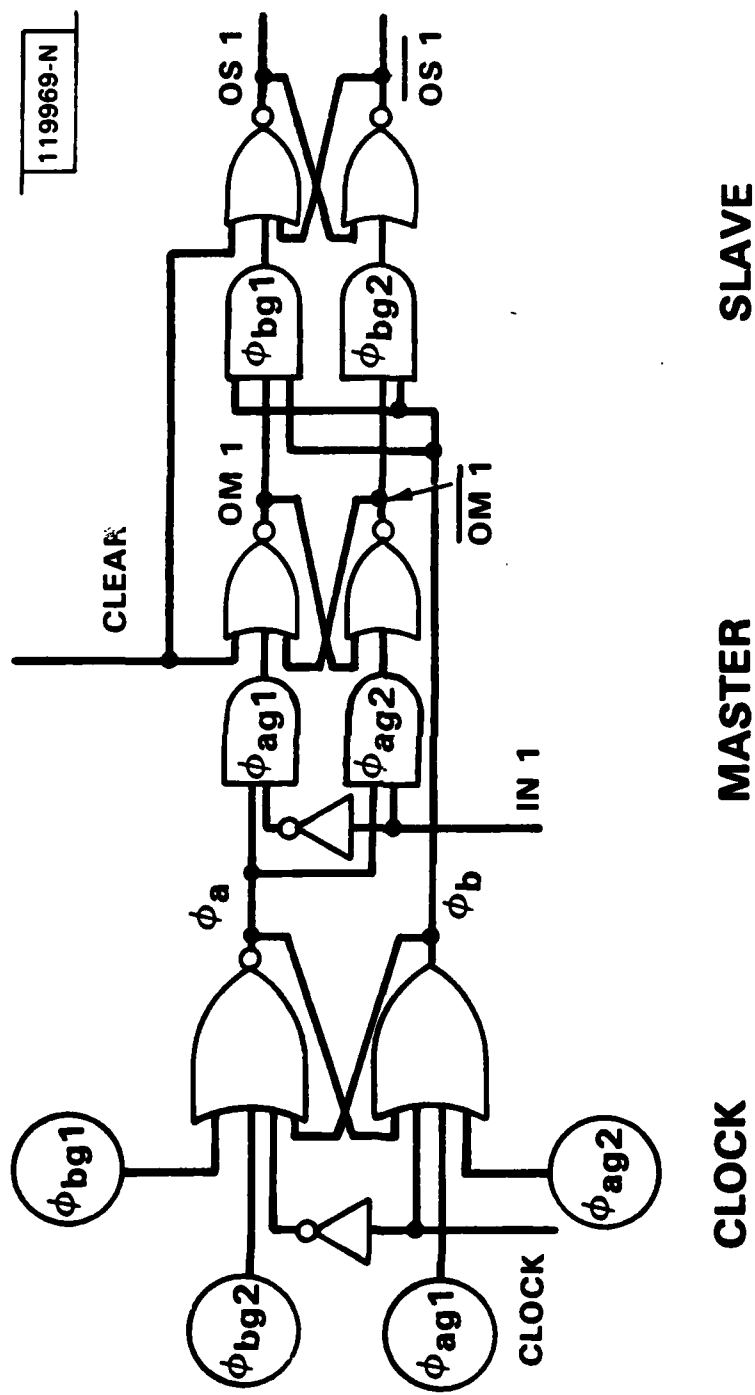


Fig. 1. Master-Slave flip-flop.

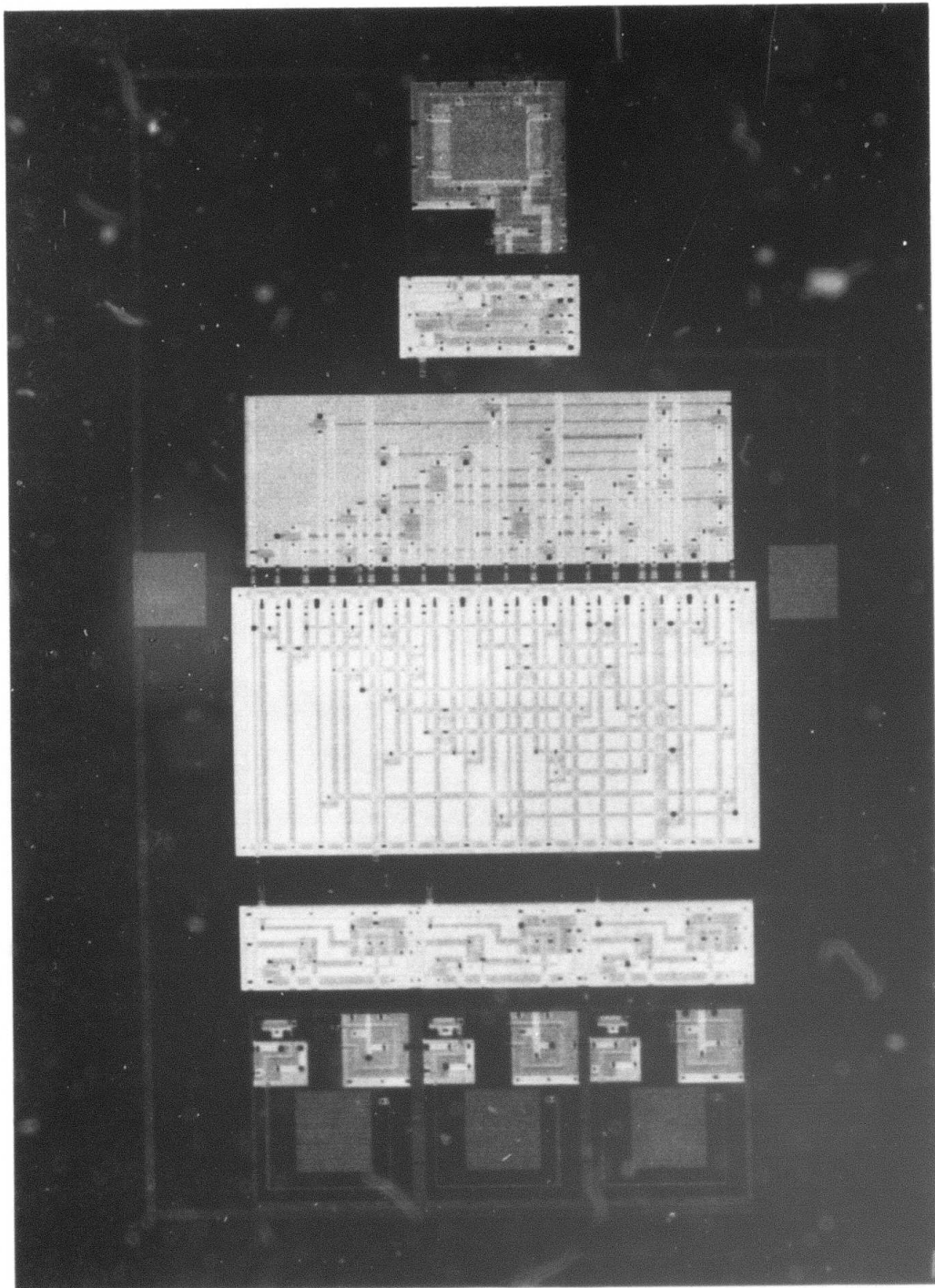


Fig. 2. Layout Master-Slave flip-flop.

the specification in a file <file-name>.lbs. Issue the command

```
lbs <file-name>
```

This will create a cif file <file-name>.cif which can then be operated on by the standard cif oriented tools (node extractors, rules checkers, display programs, etc.). It will also create a file <file-name>.stat with information about the number of gates and so on. If you wish to invoke the simulator, issue the command:

```
lbs <file-name> sim
```

If you wish to simulate a design without creating cif, issue the command:

```
lbs <file-name> sim nocif
```

7. THE SIMULATOR

A simulator has been written for use with smart terminals like the Concept 100s. When invoked, the simulator will clear the screen and present the input and output signals on the screen. Named input signals appear in the left-hand column, named output signals appear in the right-hand column. Next to each signal name is its current value, initially unknown. A long prompt will appear at the bottom of the screen. When the cursor is next to the prompt ">", you may enter a command.

Useful commands are:

- q Quit the simulator.
- 1 Read the remainder of the line for name arguments (see next paragraph). All the specified signals will be set to the value 1. If the simulation is in autopropagate mode (default, also see the P command), then the display will be updated as per the new values.
- 0 Read the remainder of the line for name arguments (see next paragraph). All the specified signals will be set to the value 0. If the simulation is in autopropagate mode

(default, also see the P command), then the display will be updated as per the new values.

- z** Read the remainder of the line for arguments (see next paragraph). All the specified signals will be undefined (hi-impedance, hence z). If the simulation is in autopropagate mode (default, also see the P command), then the display will be updated as per the new values.
- p** Propagate the input signals to the output. This will occur irrespective, and without effecting, the autopropagate mode (see the P command).
- P** Toggle the autopropagate mode. If the autopropagate mode is on, then each change to an input will be immediately reflected in the display. If the autopropagate mode is off, then propagation is delayed (and the display is frozen) until a p command is issued.
- T** Toggle the trace mode. When the trace mode is on, the display presents intermediate value information as signals propagate through the network. Trace mode assumes unit gate delay for all gates, a poor, though simple assumption. The trace mode is initially off.
- s** Save the current simulation state in the file <file-name>.sim. If the user appends an argument to the s command, the current simulation state is saved in <argument>.sim.
- g** Get the simulation state from the file <file-name>.sim. If the user appends an argument to the g command, the simulation state is loaded from the file <argument>.sim.

The name arguments are any named input or output signals appearing in the display. In addition, an argument of I will be expanded to all input signals, and O to all output signals. These letters (and G and A) are not recommended signal names. In addition, other commands and information are available to LBS experts.

REFERENCES

- [1] C. Mead and L. Conway, "Introduction to VLSI Systems," (Addison-Wesley, 1980).
- [2] J. M. Siskind, J. R. Southard, and K. W. Crouch, "Generating Custom High Performance VLSI Designs from Succinct Algorithmic Descriptions," Proc. Conference on Advanced Research in VLSI, January 1982.
- [3] A. Weinberger, "Large Scale Integration of MOS Complex Logic: A Layout Method," IEEE JSSC, SC-2, pp. 182-190, December 1967.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESD-TR-82-087	2. GOVT ACCESSION NO. A120 999	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) LBS — Lincoln Boolean Synthesizer		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER Technical Report 622
7. AUTHOR(s) Jay R. Southard, Antun Domic, and Kenneth W. Crouch		8. CONTRACT OR GRANT NUMBER(s) F19628-80-C-0002
9. PERFORMING ORGANIZATION NAME AND ADDRESS Lincoln Laboratory, M.I.T. P.O. Box 73 Lexington, MA 02173-0073		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Proj. No. 2D30 Prog. Element No. 61101E ARPA Order No. 3797
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209		12. REPORT DATE 1 September 1982
		13. NUMBER OF PAGES 16
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Electronic Systems Division Hanscom AFB, MA 01731		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES None		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Silicon Compiler VLSI IC Synthesis Boolean Logic Layout		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The Lincoln Boolean Synthesizer (LBS) creates custom integrated circuits from boolean logic equations. The currently supported integration technology is a fully customizable CMOS process with 5 micron channel width, P-tub and, one level of metal. This document describes the capabilities of LBS and provides essential information for operating LBS and an LBS simulator which has also been developed.		

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)